3D AffordanceNet: Building an Affordance Segmentation Pipeline

High Level Overview

How do we teach a system to understand how 3D shapes can be used?

Designers and developers still have to manually define and script for gameplay interactions like “this surface is sit-able” for “this handle can be pulled” → This process can be slow and inconsistent and does not scale across large environments → more time costly, large games with an immense amount of objects (they aren’t all interactable because it would simply take too long to manually label everything)

Current research focuses either on:

text-to-model or image-to-model, i.e. can identify what an object is but not what you can actually do with it OR

model-to-affordance, i.e. breaking a model down and labeling its constituent parts in order to map affordances onto it

We propose to join these procedures into a pipeline whereby we can create a plug-and-play solution for game development engines like Unreal-Engine and Unity to be able to generate and map affordances to game objects.

Why Do This?

Developers manually define every interaction

- Every “sit” “grab” or “open” zone must be hand modeled and tagged.
- This slows production, especially in large or procedural worlds
- Manual affordance tagging is inconsistent and error prone- two chairs may have two completely different interaction setups
- Artist and level designers spend hours hardcoding colliders and triggers that can be inferred directly from shape

The result: games have fewer interactable objects → worlds feel less rich and reactive

Smaller studios cannot afford teams of designers to tag and test affordances across thousands of assets.

Our intervention introduces:

- Speed: Automates affordance tagging → hours of setup into seconds
- Scalability: works across massive or procedural game worlds
- Accessibility: Creates consistent, readable affordances. Makes advanced affordance systems available to smaller or indie teams without massive resources. Closes the gap in the industry between AAA and indie games.
- Design intelligence: Lets geometry itself communicate function → bridging perception, design, and playability

New Possibilities for Game Design

NEW POSSIBILITIES AND OPPORTUNITIES IF WE SUCCEED

- Living Worlds → every object becomes intractable and meaningful
  - Worlds feel “alive” → players can touch, sit, grab, open and affect their surroundings
- New Game Genres
  - Emergent simulation, adaptive storytelling, and dynamic environment design become possible
  - Designers can prototype behavior driven worlds without coding every rule
- Empowering Small Studios
  - Indie and educational teams gain AAA-level interaction quality
  - Tools become accessible to anyone- no ML expertise required
- Scalable Creativity
  - Game engines evolve from static scene builders into affordance aware ecosystems
- Bridging Research & Design
  - Opens new domains at the intersection of machine perception, game feel and interaction design
  - Sets foundation for AI driven creativity in future Unreal or Unity pipelines

What we are doing and how we are doing it

We want to build a system that can take in a 3D model → decompose it and understand its functional geometry, identifying how its shape suggests possible actions → automate what the designer currently does manually in the game engine.

The goal is to let the model answer: “What can I do with this shape?” → the same reasoning designers currently do manually.

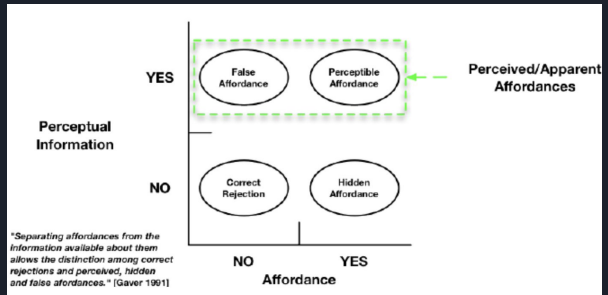
We want to automate that process through ML-driven affordance detection

- A system that takes a 3D chair asset
- Identifies geometric parts (flat surfaces, concavities, handles)
- Maps them to human readable affordances
- Outputs a mesh already tagged with extra metadata that encodes which regions correspond to which affordances→ Spawn in gameplay logic

Theoretical Grounding

Affordance Theory in Design

- Rooted in Gibson's ecological psychology - Perception is directly tied to the possibilities for action that the environment affords an agent.
- *Perception is not passive⟶ it is a direct invitation to act.*
- In design theory, this evolved into Don Norman's concept of "perceived affordance" ⟶ how form communicates function. (a flat surface invites sitting, a handle like object invites pulling)
- Gaver's work on perceptual information and affordance



Affordance relative to Perceptual Information adapted from Gaver (1991)

In Game Design

- Affordances in games are given either diagetically, i.e. through texture, form and structure of game objects, or extra-diagetically, i.e. UI cues
- Game designers manually craft these cues as object behaviors the player can see (e.g. highlighting ledges you can climb, seats you can use, or objects you can push, but these are hand authored and static)

Affordances in Games→ games rely on affordances→ how players perceive what they can do in an environment

False Affordances

The design suggests action but the game doesn't actually allow it. This creates frustration.

- Fake doors in New pokemon game
- Windows in Cyberpunk 2077 that look openable but are purely decorative
- Rope ladders in Elden ring that look climbable but are just background geometry.

Player believes action is possible- game denies it.

Hidden Affordances

- The action is possible but the player does not realize it because the cues are unclear or absent
- Ex: Ladders in Skyrim- many ladders are modeled but not climbable, but some mods make them usable. Players can't tell which ones afford climbing
- Some physics props in Half-Life 2 are throwable or breakable,but until you pick one up, you might not realize it…

Obvious/ Perceived Affordances

- Affordances that are both real and clearly communicated to the player
- The object truly allows the action and the player knows it
- Ex: Campfire in Dark Souls - glowing aura and sit prompt clearly invite resting

Deceptive Affordances

Some objects switch Affordance states–

EX:

- After proximity → the door states “press X to interact” only after you are super close to it
- Or after an event is completed

This creates dynamic Affordances- context dependent possibilities

Doors in *Bioshock* — some open later after narrative triggers → affordance changes with state.

Research Framing

Our ML model aims to learn affordances from geometry→ mapping structure → function → interaction

Why this Matters:

Current research (PartAfford, Style2Fab) stops at segmentation or visualization

We extend that into interactive functionality→ connecting how a model looks to what it can actually do in game world

This closes the gap between AI perception and designer intention→ automating one of the most labor intensive parts of 3D game production.

The Broader Research Landscape

Multiple AI research tracks already explore individual pieces of this pipeline, but none connect them for embodied, game-ready 3D interaction.

Text→ Image/ 3D generation

- Generate content from language prompts.
- OpenAI Sora, Runway Gen-2
- Focused on visual realism, not functional geometry or interaction

2D → 3D reconstruction

- Rebuild 3D models from photos or scans
- NeRF (Google), Instant-NGP (NVIDIA), Stanford 3D Vision Lab
- Reconstructs shape, but does not understand what parts “do.

3D object understanding

- Segment or classify 3D geometry.
- PointNet++, DGCNN, MeshCNN, SubdivNet, PartNet dataset (Stanford)
- Knows structure, not function. Labels parts, not actions.

Image Recognition (2D)

- Detect objects, poses, actions.
- ResNet, CLIP, SAM (Meta)
- Works in 2D pixel space no awareness of physical 3D affordances

Procedural Generative Design Tools

- Automate design variation and customization.
- Style2Fab (MIT), ShapeAssembly (Stanford), Autodesk Dreamcatcher
- Optimize shape aesthetics, not interaction semantics.

Affordance Learning

- Predict functional parts (“sit,” “grasp,” etc.)
- 3D AffordanceNet, PartAfford (Zheng et al. 2023), SegAffordSplat (ECCV 2024)
- Stops at research visualization no integration into game engines.

Project Goals

Short Term (this semester) → research prototype

_____

Build a proof of concept pipeline that shows how geometric structure predicts functional affordances

Function and Scale

- 4-5 simple furniture assets (ex: chair, table, bed , sofa, lamp)
- Train test on small subset of PartNet (90 gb) dataset
- Emphasis: conceptual clarity, segmentation quality, Unreal compatibility

Core Outputs

- Baseline model trained to predict per-face affordances
- Mesh artifact with labeled faces or materials
- Offline Python inference script (runs on NYU HPC)
- An offline script or Blueprint prototype that visualizes those affordances in Unreas

Dataset Overview

Dataset: Stanford Partnet v0

- CAD quality meshes (.obj) per object, aligned point clouds (sampled from same meshes), and hierarchical part annotations (JSON trees e.g. chair → {seat, back, leg 1, leg 2…}
- Categories: chairs, tables, lamps, storage, beds, etc
- Purpose built for part level understanding→ good for our affordance reasoning on geometry

Fits our goal

- Easily accessible and open-use for the academic community
- Clean CAD topology→ easier mesh point alignment
- Part trees mirror functional structure which is a natural bridge to affordances
- Comes with both meshes + point clouds → supports mesh, point, or hybrid pipelines

Good starting point. → once our system is validated we can extend to make it more stylized (fine tune the model) to test robustness and make it industry relevant

Current Steps

Scoping which data is essential for our ML + Game Engine goals

- Deciding minimal feature set: Mesh features Vs Point features
- Decide 5 categories to start with

Researching Affordance Label Mapping

- Define label mapping part→ structural role → affordances
  - seat → flat/stable → **sit**
  - tabletop → flat/support → **place**
  - backrest → vertical support → **lean**
- **Plan preprocessing scripts for data cleaning**

Learning HPC

Technical Progress So Far

1. Initial Exploration (more detail)
    a. Key lit review on Design Theories, geometric segmentation, various case studies
2. Model Research
    a. Evaluating 3D deep learning architectures
    b. Mapping pros and cons for each
3. Dataset selected / process of being downloaded
4. Github setup

Current Challenges

Decision Point: Mesh vs Point Cloud vs Hybrid

Dataset handling → 90 GB requires HPC access

- Offline pipeline vs unreal native → maximum flexibility across various assets

-*How expensive is this computationally

-Is our dataset robust enough

Clarifying our direction

For this semester→ achievable and pushable

Stay focused on “Can we learn affordance-relevant geometry from 3D models and project it back onto meshes for use in game engines?”

Multiple valid paths with different tradeoffs, both can serve end goal depending on semester scope! Let's break it down!

Possible routes

Point Cloud (ex DGCNN)

- Best for Early stage proof of concept
- Simple and Fast → if we want speed,
- Needs projection back onto Meshes
- Good starting point→ can evolve
- No explicitly topology
- Fuzzy
- Can mislabeled edges
- No direct surface normals

Hybrid “mesh aware point learning” (DGCNN → Mesh)

We train model in flexible robust point cloud domain and then project and refine results onto meshes→ either using classical geometry ops or a small refinement GNN later

- Best for production pipeline (after baseline works)
- Moderately complex
- Highest fidelity and flexible
- Ideal long term system
- Requires synchronization between point and mesh data
- Engineering complexity
- Longer training pipeline - 2 stages

Mesh-native (ex: MeshCNN/ DGNET)

- Best for Research grade geometry reasoning
- COMPLEX
- 1:1 topology aware
- Great for long term integration and later refinement
- Topology dependent (bad with generalization)
- High preprocessing cost
- Heavy GPU
- Spase with real world applicability
- Small data regime issues
- In theory they produce perfect topology aligned labels but it takes long and you can get even more exact/ generalized and faster results in the same period of time with the Hybrid method

Hybrid Point→ MEsh Affordance Pipeline

Goal: Train a model that learns affordances from geometry – mapping shape → function → interaction – and outputs labeled meshes ready for engine integration

Why Hybrid (not pure mesh)--> Mesh native GNNs (MeshCNN, DGNet, etc) are topology faithful but slow, brittle, and hard to scale

Pointbased models are stable, fast, and handle stylized or scanned assets.

The hybrid pipeline combines both strengths: learning in point-space, delivering results in mesh space.

Overview: (open for change and suggestions)

1. Input Prep
   a. Start from PartNet meshes +point clouds with part labels
   b. Define a part→ affordance mapping (seat → sit, handle → grasp)
2. Dense point Sampling (optional since we have pointcloud sets)
   a. Resample if we find we need finer details for curved or complex objects
   b. Can downsample for lightweight debugging
3. Training (DGCNN or other model)
   a. Train a dynamic graph CNN on point clouds for supervised affordance segmentation
   b. Learns local geometry patterns (flat → sit, concave → contain)
4. Projection to Mesh
   a. Run inference on new assets
   b. Project per point predictions back onto mesh faces using methods from papers (nearest neighbor or barycentric interpolation)
   c. Result: mesh with per face affordance labels.
5. Game engine integration (what we are deciding)
   a. Offline or In real time
   b. IF offline, import labeled mesh into UNreal, Editor Utility Blueprint reads labels→ auto assigns gameplay tags and colliders
   c. Assets become affordance aware on import

Breakdown of Static vs Dynamic Affordances→ Realtime versus Offline Preprocess

Offline:

- Runs python / HPC environment
- Model runs on mesh data -> exports labeled .fbx or .json
- Simple (use editor utility blueprint to read metadata)
- No performance cost at runtime
- Trains on full 90GB easily on HPC
- Cross engine compatibility (works with unreal Unity Blender)
- Shorter Dev time (good for our starting projects)
- Clear Academic deliverable: “smart mesh tagging”

Designer Workflow: Designer imports pre tagged assets; can edit in engine

- Static affordance behavior (does not change after export)
- Transparent (for debugging)

Best for asset preprocessing, pipeline automation

In Engine Pipeline

- Directly inside Unreal or Unity (editor or runtime)
- Model runs automatically when importing or simulating assets
- Complex- ONNX TensorRT model import, or C++ plugin
- Higher performance cost- live inference during gameplay or import
- Limited scalability- engine environment is not built for large scale ML
- Engine specific plug in (unreal first, unity separate build)
- Long development time (multi semester maybe PhD?)

Designer workflow: designer drags in raw assets; engine auto generates tags on the fly

Dynamic- can adapt possibly as world or physics changes (tipped chair loses sit)

Ambitious for a demo

But best for real time interaction, adaptive environments

Long term goals

Extended Version (thesis or into next year) Long term

_____

Long term project evolves this into a full geometry aware authoring system→ An Ai assisted authoring system that can analyze 3D assets and automatically tag, colliderize, and connect gameplay logic in Unreal and Unity

- Requires a reliable ML backbone that can infer affordances (what we are making now)
- **Geometry to function → model predicts affordance regions on new meshes (python HPC environment)**
- **Authoring Layer → tool lets designers edit refine or override predicted tags (Unreal editor utility)**
- **Integration - gameplay systems read these takes to attach collisions , triggers, or AI interactions (Unreal Runtime)**
- **Future experiments can involve comparing mesh only vs pointclouds and hybrid representations**
- **Test generalization to new asset categories**
- **Evaluate contiguity, consistency, and engine usability of predictions**
- **We want our system ot understand geometry well enough to automatically attach the right gameplay tags, colliders, and interaction triggers→ saving hours of manual labor**

Next Steps

Confirm which modality we want to train on first

Dataset download - start looking at the Data- researching preprocessing of data + role mapping

Discuss feasibility of hybrid pipeline within timeline

Access NYU HPC - review setup and resource requirements

Finalize model research and selection→ once we pick dataset route

Continue documenting process for publication and collaboration

Questions for discussion

- Offline vs online unreal
- dataset
- What would I need for materials and computing
    - Get into more after understanding NYU HPC

References

Text References (refer to our Texts doc)

- James J. Gibson – The Ecological Approach to Visual Perception
- Don Norman - The Design of Everyday Things
- Steve Swink - Game Feel: a Designer's guide to Virtual Sensation
- Contemporary ML Cast Studies
  - PartNet
  - 3DAffordanceNet
  - PartAfford/ SegAffordSplat
  - Style2Fab
- Websites on Machine Learning Pipeline deployment
  - https://www.mdpi.com/2079-9292/13/10/1888
  - https://medium.com/@Sanjaynk7907/batch-vs-real-time-machine-learning-understanding-the-processing-powerhouse-and-the-speedy-bb7f962b9be6
  - https://elib.dlr.de/211386/1/MLEnabledSystemsModelDeploymentMonitoring.pdf
  - https://www.metricpanda.com/rival-fortress-update-2-preprocessing-and-bundling-game-assets/
  - https://www.hopsworks.ai/post/mlops-to-ml-systems-with-fti-pipelines
- Papers on ML techniques
  - https://arxiv.org/pdf/2412.10977?